

Figuring out Actors in Text Streams: Using Collocations to establish Incremental Mind-maps

T. Rothenberger, S. Öz, E. Tahirovic

Johann Wolfgang Goethe-University Frankfurt am Main, Dept. of Computer Science
Robert-Mayer-Str. 11-15, 60486 Frankfurt am Main, Germany
Email: {tahirov, oez, rothenb}@cs.uni-frankfurt.de

C. Schommer

University of Luxembourg, Dept. of Computer Science and Communication
6, Rue Coudenhove-Kalergi, L-1359 Luxembourg
Email: christoph.schommer@uni.lu

March 20, 2008

Abstract

The recognition, involvement, and description of main actors influences the story line of the whole text. This is of higher importance as the text per se represents a flow of words and expressions that once it is read it is lost. In this respect, the understanding of a text and moreover on how the actor exactly behaves is not only a major concern: as human beings try to store a given input on short-term memory while associating diverse aspects and actors with incidents, the following approach represents a virtual architecture, where collocations are concerned and taken as the associative completion of the actors' acting. Once that collocations are discovered, they become managed in separated memory blocks broken down by the actors. As for human beings, the memory blocks refer to associative mind-maps. We then present several priority functions to represent the actual temporal situation inside a mind-map to enable the user to reconstruct the recent events from the discovered temporal results.

1 Motivation

The main idea is to try to differentiate sentences on the basis of actors that appear in them. We pursue *one actor per sentence* rule to avoid ambiguity, so that each sentence becomes a part of the knowledge base for just one actor. Each actor who appears once receives his own mind-map. We reconstruct the story line of the text from these mind-maps to find out more about single actors by examining interaction among them. Because of an incremental nature, we examine one sentence per time in a

subsequent manner while updating our mind-map continuously. In order to understand the motivation and the subject itself, it would be helpful to take some time to closer examine the corresponding concepts. Some of the concepts are subject of research in computational linguistics while others are more related to computer science. We interpret *incremental* as we refer to its ability to gradually adjust to changes that happen over a period of time so that the present condition reflects these changes.

The central concept of this work is the concept of collocation, which is one of the most frequent stipulated concepts in natural language processing and computational linguistics. The difficulty to reconcile all aspects of collocations in one single definition has occupied many natural language experts and computer linguists over a large period of time. In [2], The term *collocation* . . . is used for word combinations that are lexically determined and constitute particular syntactic dependencies such as verb-object, verb-subject, adjective-noun relations, etc. The mind-map concept we would like to present in this paper bears resemblance to canonical concept of the associative mind-map, in a sense that the derivation a of mind-map address, for a particular collocation, results from the collocation's structure/formation itself. Questions similar to our task have already been raised in the domain of computer linguistic particularly with regard to *information* and *discourse* structure.

Most the work that has been done on this field is to present the semantical, temporal and psychological attributes of a discourse in a way that is interpretable by a machine. This differs from our task in the sense that features of our discourse analysis will be presented to a human user who will be hopefully able to get a good picture of the discourse considered. In [1], a topic-tree model to form a discourse structure acceptable for further machine interpretation is introduced. They denote the focus of a sentence as a topic that corresponds corresponds to our notion of *actor* or *character*, who has to be identified for each sentence. In [4], an associative mind-map is established that bases on artificial cells that communicate to each other and merge in case they represent the same content. Connections through other artificial cells are done by the Hebbian Learning principle, taking into account a forget in case the connection is less stimulated.

2 Architecture

Given a text stream, we will identify the main actors in order to understand how they interact and concentrate on current events that have taken place in the shorter past. The interpretation of the concept *shorter past* will be defined by a function in dependence to the position of the collocation in the text and the number of times where this collocation appears in the present moment. A detailed elaboration of the priority functions mentioned follows. The interaction between main actors can often reveal a coherence about the story line of the text. This brings us to the incremental nature of our work that permits an insight into the story line at an user-defined point of time. Each sentence has to pass a pre-processing step, which is done sentence-wide. Here, the sentence is brought to a form that supports the identification of the corresponding

actor/character.

2.1 Initializing the Mind-Map

After having successfully identified the actor, the information in the sentence becomes part of the mind-map that is set up for this particular actor; additionally, we keep in mind the information about a story line that is broken down by actors in the text. The user interactively may decide the direction of action in the text. He will be presented the most newsworthy entries of the mind-map according to the already mentioned priority function, for all or for a couple of chosen actors. In this way, we can spare the user from reading the whole text in search for a particular spot or in his efforts to get an understanding of what the text is about. The system is validated on fairy tales as these texts are simple regarding the linearity of the story line and the structure of the sentences. In particular, the tests and validation of the results will be carried out on the text of *Little Red Riding Hood by the Brothers Grimm*.

Before we manage collocations in the mind-map of a particular actor, each text stream item is pre-processed. This does not violate the request of a stream as the collocation representation in the mind-map follows immediately. For this, we perform several pre-processing actions like stemming and parsing. This is not of too much freedom in rearranging the input stream. A more efficient identification of actors can be achieved if the text stream can be modified at once, therefore phenomena like main clauses and subordinate clause, direct/reported speech, active/passive etc. are processed. We stem each word and identify each compound sentence while breaking them down into a more simple representation. With loss of information, we dissolve multiple sentences by deleting conjunctions and rely on some corrective actions by the human supervisor in the sense of human-computer interaction. However, for a directed or reported speech, it would be advisable to embed the statements that appear between quotations into the story. This could be accomplished by erasing quotation marks and omitting the notation about the speaker. Interrogative sentences as such carry no information relevant for our task, because it is quite difficult and therefore hard to determine: what is their contribution in forming the information structure of the text? For this, we believe to be wise to leave them out as a part of the pre-processing step.

To discover a unique form of the sentences and to transfer indirect sentences, we use again a stemmer, a particular grammar and the human supervisor who takes some corrective actions and who detects the correct reference between pronouns and the corresponding actors. Additionally, we manage the position of the sentence/collocation in the text by a simple counter that is incremented sentence by sentence. The following is an example, how the preprocessing step looks like. Following an original text like

→ Der Wolf legte sich wieder ins Bett und fing an zu schnarchen. Der Jäger ging vorbei. Er dachte, die alte Frau schnarcht so laut, da muss ich einmal nachsehen. Er trat ein. Im Bett lag der Wolf, den er so lange gesucht hatte.

we dissolve the compound sentences incrementally. Word items are brought to their

basic form, categories are assigned, and the correct references between pronouns and actors established.

- t_1 : Der Wolf legte sich wieder ins Bett. \Rightarrow Wolf(N) - legen(V) - Bett(N)
- t_2 : Der Wolf fing an zu schnarchen. \Rightarrow Wolf(N) - anfangen(V) - schnarchen(N)
- t_3 : Der Jäger ging vorbei. \Rightarrow Jäger(N) - gehen(V) - vorbei(N)
- t_4 : Er dachte. \Rightarrow Frau(N) - sein(V) - alt(ADJ)
- t_5 : Die alte Frau schnarcht so laut. \Rightarrow Frau(N) - schnarchen(V) - laut(ADJ)
- t_6 : Ich muss einmal nachsehen. \Rightarrow Jäger(N) - nachsehen(V) - Haus(N)
- t_7 : Er trat ein. \Rightarrow Jäger(N) - eintreten(V) - Haus(N)
- t_8 : Im Bett lag der Wolf. \Rightarrow Bett(N) - liegen(V) - Wolf(N)
- t_9 : Er hatte den Wolf so lange gesucht. \Rightarrow Jäger(N) - suchen(V) - Wolf(N)

2.2 Assign collocations to an actor

To assign each sentence to one particular actor, the canonical sentence/collocation structure is made up of several parts:

- A sentence starts with a subject representing the actor (to whom the sentence is assigned).
- It is followed by a verb in the sentence that indicates the relation between the actor and the third part of the collocation.
- The third part is a generalized notion of an object in the sentence, which is involved in some way with the actor. It is normally a noun or an adjective depending on the way, how the sentence/collocation in question is generated in the pre-processing.
- At the end of each collocation, a position counter supports the process of representing collocations in a correct order.

The collocation that ends with an adjective is created from a word combination *adjective + noun* in the original text. The *adjective* describes one particular feature of the *noun*; this yields on a new collocation in which *noun* becomes the *actor* and *adjective* an *object* of the collocation. We assign the word *sein* a linking role in the collocation and represent it as a fact. The new collocation inherits the collocation number of the sentence, which it was a part of and thus preserves the time flow of the text. This way we achieve the unique sentence structure that we need, and keep the original semantical value of the text.

$$\begin{aligned} \textit{alte Frau} &\Rightarrow \textit{Frau} - \textit{sein} - \textit{alt} \\ \textit{Blumen standen ringsherum} &\Rightarrow \textit{Blume} - \textit{stehen} - \textit{ringsherum} \end{aligned}$$

We keep a *dynamic list* of each *actor* who appears in text. Each *actor* has one main mind-map block and one priority list for each priority function. Priority lists save collocations sorted by respective priority function enabling output in accordance with the particular function. These functions depend, in general, on the present moment and the repetition behavior of a particular collocation. In this respect, our strategy is as follows:

- We identify the *actor* in the currently processed collocation.
- We prove if this is the first appearance in the text. This can be achieved by contacting the dynamic collective *actor* list. If so, a new mind-map block is allocated for this *actor*. If the actor already exists in the mind-map, we then use the *verb* (second part) of the collocation as a key to save this collocation in his main mind-map block. It is possible that this *verb* appears in relation with this *actor* again: then a list for it already exists in the mind-map. Such a list is kept for each *verb*: what we still have to do is to examine if the *object* (third part) can be found in the list. If so, we detect a reoccurrence of the current collocation.
- Further steps are to re-compute the priority functions and to sort the priority lists according to the new values. We preserve this way the correct order in the lists which we later use in output for chosen actors.

2.3 Priority Functions

The priority lists is managed for each *actor*. The presentation of the results - showing recent/current collocations broken down by actors - occurs directly from these priority lists. Each priority list has its own priority function by which the collocations become sorted. Dependent on the user's decision (who may intervene time by time), items of chosen actors are exported from the priority lists. However, not all collocations can be presented as a result. Since we try to model the current events in the story line, we prefer of just taking the most *recent* collocations.

To provide the user with insight into each actor throughout the entire story line, we present at least five entries. Additionally, we use a threshold Δ to allow the output of the most recent collocations according to the priority function and display those entries in the actor's priority list that lie above. The results for a particular actor are presented in a separate window, whereas the font size in which the results are written depends on the value evaluated by the priority function.

With the idea of priority functions, we model a concept of oblivion in the program. Taking the values computed by priority functions, we obtain an order among the collocations. For further elaboration it is important to define the notion of *recent* or *current* events. Observing repetition of a particular collocation could increase the importance

of the collocation for the story line. Besides chronological features of the collocations, we considered the number of repetitions as a meaningful feature in determining the priority of the collocation. This means that the collocations which have several occurrences in the text, should receive a larger priority than those which occur only once. Of course there are many other features which could seem reasonable in calculating the priority of the collocation, like appearance of a main character in it. We incorporated the impact of repetitions in the calculation of two of our priority functions. All priority functions should share some important properties in order to model the story line primarily in accordance with its chronological aspects. Such a function must be defined on a subset D of the set of natural numbers N . It must be continuous and monotonously decreasing:

$$\forall x, y \in D : x < y \Rightarrow f(x) > f(y)$$

We provide several priority functions from which the user can choose: let $\vec{x}^k \in N^d$ be a dynamic vector of all sentence numbers in which the collocation k appears, d is at most the number of all sentences in the text

$$F_1(c, \vec{x}^k) = \sum_i 0.5^{c-x_i^k}$$

where c the number of the sentence which is currently examined. We can see that this function incorporates a repetition characteristic of a collocation in the calculation of the priority. The most recent occurrences of the collocation carry more weight in the sum than those which lie far in the past. Because we chose the geometric progression the earlier occurrences will be *forgotten* very fast. If a occurrence happened in a very near past it has a relatively big influence on the whole sum. The coefficient of the geometric progression remains 0.5 all through the process.

The second priority function considers the number of repetitions only (d the dimension of the vector \vec{x}^k of the first function). In this case we increase with each repetition the coefficient of the geometric progression. The priority is calculated in dependence of the current coefficient \hat{a} , number of the currently examined sentence c and the number l_k of most recent sentence in which a collocation k appeared:

$$\hat{a} = \sum_{i=1}^d 0.5^i$$

Since we increase the coefficient, with each repetition, priority of the collocations that experience repetitions is uprated. We can see that the step of increase for the coefficient is calculated as a sum of a geometric progression with a start coefficient 0.5. This function rates repetitions higher than the first one since with each repeated occurrence the coefficient changes permanently and the chronological order of repetitions plays no part any more. The second function is then:

$$F_2(\hat{a}, c, l_k) = \hat{a}^{c-l_k}$$

The third function is the simplest of all three. The repetitions are considered irrelevant. Priority depends on the current sentence number c and the number l_k of the last sentence with a particular collocation k observed.

$$F_3(c, l_k) = 0.5^{c-l_k}$$

2.4 Example

These functions provide a way to measure relevance of repetition for calculating the priority of collocations. Let $c = 20$ and

- *Wolf - sein - böse* (fifth sentence)
- *Wolf - sein - böse* (fifteenth sentence)
- *Wolf - sein - böse* (seventeen sentence)

then

- $F_1 = 0.5^{20-5} + 0.5^{20-15} + 0.5^{20-17} \approx 0.15628$
- $F_2 = (0.5^1 + 0.5^2 + 0.5^3)^{20-17} \approx 0.66992$
- $F_3 = 0.5^{20-17} = 0.125$

We can already see on this example how repetitions exert a different impact on the respective function evaluation.

3 Implementation

Figure 1 shows the workbench of the software system, which has been already presented in [3]. It consists of several panels: in the top, the user may select the discovered actors, which may be human beings, references to human beings like *Jedermann*, or even animals. We then can select among the different priority functions receiving the mind-map with the associative relationships, including a relevance value. In this respect, Figure 1 shows the situation for the actor *Blumen* after having read the text stream. The highest association is with *sein schön*. The priority values depend on the chosen priority function.

4 Test and Validation

As an example, we examine the german fairy tale *Rotkäpchen* by taking the priority functions for each actor. With c , we express the current sentence number. In Figure 2, we compare first the impact of unequal influence of repetition for two priority functions of the same actor. The collocation *Blumen - sein - schön* appears both in fifteenth as well as in twentieth sentence with

$$F_1(21, (15, 20)) < F_2(0.75, 21, 20)$$

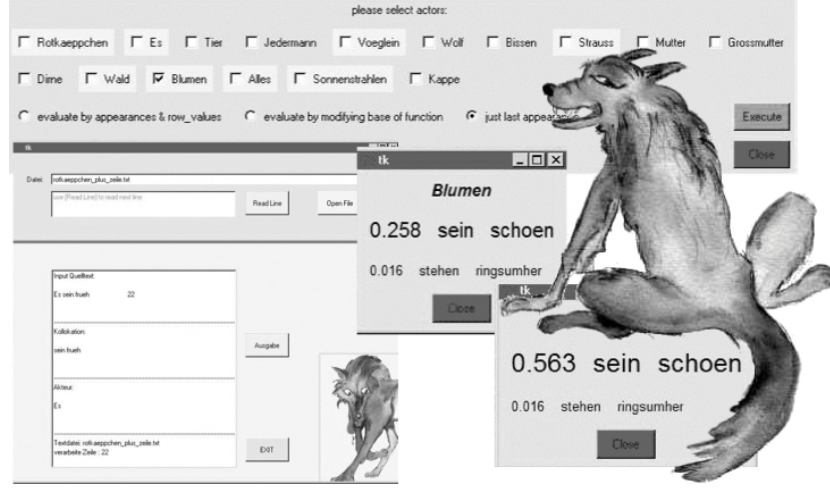


Figure 1: The situation for the actor *Blumen* after having read the text stream. Depending on the priority function, the values differ, but the highest association keeps stable ([3]).

Blumen (F_1)			Blumen (F_1)			Blumen (F_1)		
Priority	Verb	Object	Priority	Verb	Object	Priority	Verb	Object
0.516	sein	schön	0.258	sein	schön	0.0	sein	schön
0.031	stehen	ringsumher	0.016	stehen	ringsumher	0.0	stehen	ringsumher
Blumen (F_2)			Blumen (F_2)			Blumen (F_2)		
Priority	Verb	Object	Priority	Verb	Object	Priority	Verb	Object
0.75	sein	schön	0.563	sein	schön	0.32	sein	schön
0.031	stehen	ringsumher	0.016	stehen	ringsumher	0.0	stehen	ringsumher
Blumen (F_3)			Blumen (F_3)			Blumen (F_3)		
Priority	Verb	Object	Priority	Verb	Object	Priority	Verb	Object
0.25	sein	schön	0.25	sein	schön	0.0	stehen	ringsumher
0.016	stehen	ringsumher	0.016	stehen	ringsumher			
$\tau(21)$			$\tau(22)$			$\tau(32)$		

Figure 2: The actor *Blumen* and its associated collocations after having read the twenty-first sentence (left). The collocation *Blumen* - *schön* proves the highest value with priority function $F_2 = 0.563$ whereas priority functions $F_1 = 0.258$ and $F_3 = 0.25$ are lower. The collocations for *Blumen* after having read the thirty-second sentence (right). The discrepancy between *schön* and *ringsumher* is strongest for priority function F_2 .

Großmutter (F_1)		
Priority	Verb	Object
0.063	sein	schwach
0.063	rufen	—
0.0	wohnen	draussen
0.0	wohnen	Wald
0.0	sein	krank

Großmutter (F_1)		
Priority	Verb	Object
0.004	sein	schwach
0.004	rufen	—
0.0	wohnen	draussen
0.0	wohnen	Wald
0.0	sein	krank

Großmutter (F_1)		
Priority	Verb	Object
0.5	wohnen	draussen
0.5	wohnen	Wald
0.125	sein	schwach
0.125	sein	krank
0.008	schenken	Kappe

Großmutter (F_2)		
Priority	Verb	Object
0.316	sein	schwach
0.063	rufen	—
0.0	wohnen	draussen
0.0	wohnen	Wald
0.0	sein	krank

Großmutter (F_2)		
Priority	Verb	Object
0.1	sein	schwach
0.004	rufen	—
0.0	wohnen	draussen
0.0	wohnen	Wald
0.0	sein	krank

$\tau(10)$
 $\tau(32)$
 $\tau(36)$

Figure 3: The history of *Großmutter* within the text at time points 10, 35, and 36. The word item *wohnen* has disappeared (value of 0.0) as it has been forgotten whereas the association with *schwach* weakly exists. The association *Großmutter* - *rufen* has no object as *rufen* does not need one.

We may observe that the second function is stronger influenced by repetitions. The priority values for *Blumen* - *stehen* - *ringsumher* are the same since this collocation occurred only once so far. In Figure 2, all three functions for the same actor here. The third function obviously calculates priority without regard to repetition since

$$F_3(22, 20) < F_1(21, (15, 20)) < F_2(0.75, 21, 20)$$

After having read the thirty-second sentence, the list of collocations has changed (see Figure 2). Following the priority function F_1 , all associated weights are 0.0 whereas priority function F_2 gives a value of 0.32 for *Blumen* - *sein* - *schön*.

A similar situation occurs for *Großmutter* (see Figure 3). F_1 forgets collocations which occurred more than once faster than F_2 . Moreover, we can recognize the geometric progression of the priorities calculated by F_1 for the actor *Rotkäppchen*. None of these collocations occurred more than once. The last of them (or the first by priority) occurred in the sentence number 32 with

$$F_1(32, (32)) = 0.5^{32-32} = 0.5^0 = 1$$

Once having read the text stream, the main memory for *Wolf* is

- gehen: [[Wald, Großmutter] [Wald, 0.6, [10, 15]] [Großmutter, 0.5, [18]]] with a priority list of
 \rightarrow sein - böse - 0.5

- sein - hungrig - 0.4
- sein - listig - 0.3
- suchen - Nahrung - 0.2
- ansprechen - Rotkäppchen - 0.1

5 Conclusions

In this paper, we have presented an architecture of revealing the action/story line in texts by recovering collocations between the actors. This has been done incrementally as the text can be seen as a text stream that once it is read it is lost. Bringing all sentences on a unique form allows the identification of the actor in a particular sentence. The number of actors per sentence is limited to one. The collocations corresponds to each involved action of an actor; this is managed by a mind-map allocated for each actor. A set of priority functions sort the collocations in each actor's mind-map by their actuality. An user may then decide which part and above all when he wants to check the story line. The output result presents the most actual collocations that actors are involved in; from these, the user may conclude how the actors interact with each other.

6 Acknowledgement

This work has been performed at the research laboratory *ILIAS - Intelligent and Adaptive Systems* within the project *TRIAS*, which is funded by the University of Luxembourg.

References

- [1] G. Heyer, M. Luter, U. Quasthoff, T. Wittig, C. Wolff: Learning Relations using Collocations. In: Maedche, Alexander; Staab, Steffen; Nedellec, C.; Hovy, Ed (ed.). Proceedings of IJCAI Workshop on Ontology Learning, Seattle/WA, August 2001, pp. 19-24.
- [2] B. Krenn : CDB - A Database of Lexical Collocations. Austrian Research for Artificial Intelligence.
- [3] T. Rothenberger, S. Öz, E. Tahirovic: INASCO - Bestimmung eines inkrementellen Assoziativspeichers für Kollokationen. Internal Report. Johann Wolfgang Goethe Universität Frankfurt am Main, 2006.
- [4] C. Schommer: Incremental Discovery of Association Rules with Dynamic Neural nodes. Proceedings of the Workshop on Symbolic Networks. ECAI 2004, Valencia, Spain. 2004.
- [5] B. Schroeder, M. Hilker, R. Weires: Dynamic Association Networks in Information Management. Proceedings 21st International Conference on Computer, Electrical, and Systems Science, and Engineering (CESSE 2007). Vienna, Austria, 2007.